

Text Segmentation with Combination of Text Categorization by String Vector based KNN

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN algorithm for automating the text segmentation, by introducing the text categorization. In the previous work, we proposed the KNN version as an approach to the text segmentation, but must present the domain as a tag, manually. In this research, we adopt the KNN algorithm which receives a string vector as the input data, as the approach to both the text segmentation and the text classification. In the proposed system, only a text without its domain tag is given as the input, it is classified into a domain by the text classification, and each paragraph pair is classified into boundary or continuance by the classifier which corresponds to the domain. This research is intended to automate the text segmentation by introducing so, and to improve the performance of both tasks, using the string vector based KNN algorithm.

I. INTRODUCTION

The text segmentation is defined as the process of segmenting a text into subtexts based on the topic transition. A boundary which indicates a strong topic transition is set between paragraphs, so the text segmentation is viewed as a binary classification of adjacent paragraph pairs into boundary or continuance. The process of text segmentation is to partition a text into paragraphs, generate adjacent paragraph pairs by sliding a two-sized window on paragraphs, and classify each adjacent paragraph pair into one of the two categories. A boundary is put on the point between paragraphs in each pair which is classified into boundary, and a text is segmented into subtexts following the boundaries. Because the classification which is mapped from the text segmentation belongs to the domain dependent classification where an even same entity may be classified differently depending on the domain, it is required to present the domain as well as a text to the text segmentation system.

This research is motivated by the requirement of presenting a domain for executing the text segmentation. It is composed with independent classification tasks each of which corresponds to its own domain. A machine learning-based classifier is allocated to each domain, so we need to know the domain in advance in nominating the corresponding classifier. The process of assigning a domain to a text is automated by the text classification. The text segmentation

relates to the text classification for presenting only a text without its domain for segmenting a text.

The idea of this research is to install the text classification into the text segmentation as its supplementary task and apply the string vector based KNN algorithm to both tasks. We define the semantic similarity between two string vectors as an operation on them and modify the KNN algorithm into a string vector-based version. Each text is classified into one among the predefined domains by the string vector-based version. The text segmentation is mapped into the binary classification of adjacent paragraphs into boundary or continuance. A boundary is put between paragraphs in each pair which is labeled with boundary, and the text is segmented into subtexts, following boundaries.

Let us mention some benefits from this research. The problems in encoding texts into numerical vectors are solved by encoding them into string vectors. The performance of the text segmentation and the text classification is improved by adopting the string vector based KNN algorithm. The process of assigning a domain to a text for the text segmentation which is defined as a domain dependent classification is automated by connecting the text segmentation with the text classification. In this research, we open the chance for modifying other machine learning algorithms into string vector-based versions.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with previous works which are relevant to this research. It is intended to connect the text segmentation with the text classification for automating the classifier selection. The direction of exploring previous works is to find cases of applying the proposed version of KNN algorithm to the text classification and the text segmentation and mentioning the Neural Text Categorizer as a typical neural model which receives a string vector.

The distinguishable point of this research is that the text segmentation is connected to the text classification for implementing the complete text segmentation system. This section is intended to explore previous works for providing background for this research.

Let us explore the previous works on applying the modified version of KNN algorithm to the text classification. In 2017, it was initially proposed that the string vector-based version of KNN algorithm should be applied to the text categorization as a position paper by Jo [4]. In 2018, the modified KNN algorithm was validated in the task by Jo [6]. The journal article which describes in detail the modified KNN algorithm and its application to the word classification is finally prepared for its publication by Jo [11]. In the pervious works, better performances were observed in applying the proposed version of the KNN algorithm to the text classification.

Let us explore the previous works on applying the modified version of KNN algorithm to the text segmentation. In 2017, it was initially proposed that the string vector-based version of KNN algorithm should be applied to the text segmentation as a position paper by Jo [5]. In 2019, the modified KNN algorithm was validated in the task by Jo [10]. The journal article which describes in detail the modified KNN algorithm and its application to the text segmentation is finally prepared for its publication by Jo, as well as text segmentation [12]. It was pointed out that the text segmentation is a domain dependent classification, and it is necessary to tag a domain to an input text automatically.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [1]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [3]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [8]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [2].

Let us mention the differences of this research from the previous works which are explored above. The text classification which exists as a single system in [12] is applied to the domain selection in the text segmentation. By introducing the text classification to the text segmentation, we automate tagging a domain to a text. Machine learning algorithms will be innovated by inventing various ones which process directly string vectors as their input. In the next research, we will consider introducing text segmentation to text classification as the reverse case of this research.

III. PROPOSED WORKS

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a text into a string vector and the proposed similarity metric. In Section III-B, we describe the string vector based version

of KNN algorithm. In Section III-C, we present the system architecture of the text segmentation system. In Section III-D, we present the system architecture of its extended version by connecting it with the text categorization system.

A. Encoding Proceresss

This section is concerned with the string vector as the text representation, and the similarity between string vectors. A string vector is a finite ordered set of strings; numerical values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [9] for studying the encoding process in more detail.

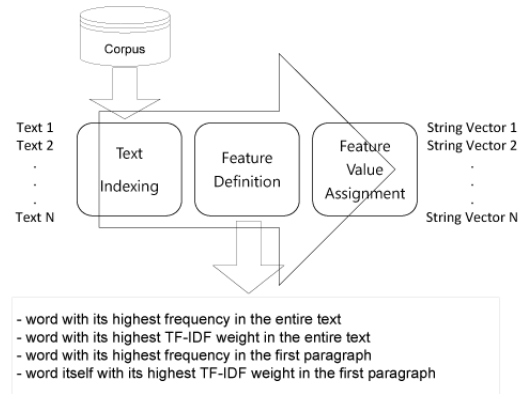


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and

$\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given as a normalized value between zero and one, as shown in equation (3),

$$0 \leq \text{sim}(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

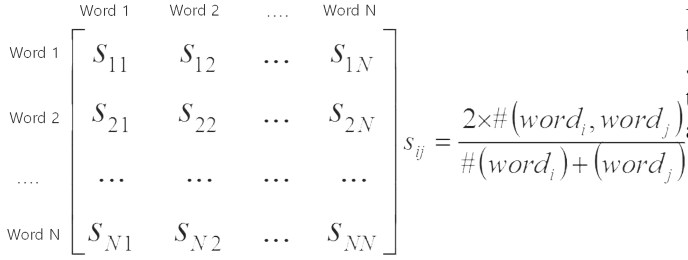


Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\text{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\text{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(t_{i1}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [7]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical

weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. A novice word is encoded into a string vector notated by str . Its similarities with the string vectors which represent the sample words are computed by equation (4), as $\text{sim}(\text{str}, \text{str}_1), \text{sim}(\text{str}, \text{str}_2), \dots, \text{sim}(\text{str}, \text{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

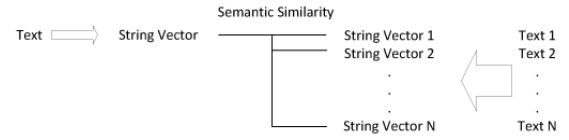


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, \text{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\text{str}_1^{\text{near}}, \text{str}_2^{\text{near}}, \dots, \text{str}_k^{\text{near}}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

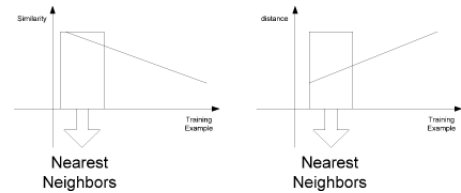


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\text{str}_i^{\text{near}})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$.

The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$\text{label}(\mathbf{str}) = \arg\max_{j=1}^m \text{Count}(\text{Nr}, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text segmentation system which adopts the string vector based KNN algorithm. In Section III-B, we described the proposed version of KNN algorithm as the approach to the text segmentation. It is viewed as a binary classification which classifies a paragraph pair into boundary or continuance. This section is intended to describe the text segmentation system with respect to its function and its architecture.

The process of gathering sample paragraph pairs and classifying a novice one, is illustrated in Figure 6. Because even a same paragraph pair may be classified differently depending on the domain, the task is called domain dependent classification. Sample paragraph pairs are gathered domain by domain, and each pair is labeled with boundary or continuance. The paragraph pairs are taken from texts which are tagged with their same domain, each paragraph pair is classified into boundary or continuance. Sample paragraph pairs which are labeled with one of the two categories are encoded into string vectors in each domain.

The entire architecture of the proposed text segmentation system is illustrated in Figure 7. A text is given as the input, and adjacent paragraph pairs are extracted by partitioning the text into paragraphs and slide the two sized window on them. The sample paragraph pairs in the boundary group and the continuance group and ones which extracted from the text are mapped into string vectors in the encoding module. The paragraph pairs from the text are classified into one of the two categories in the similarity computation module and the voting module. A boundary is put between paragraphs in

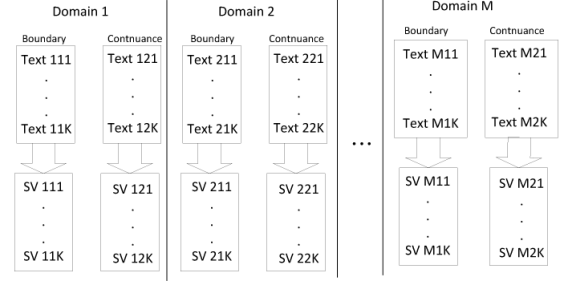


Figure 6. Preparation of Training Examples

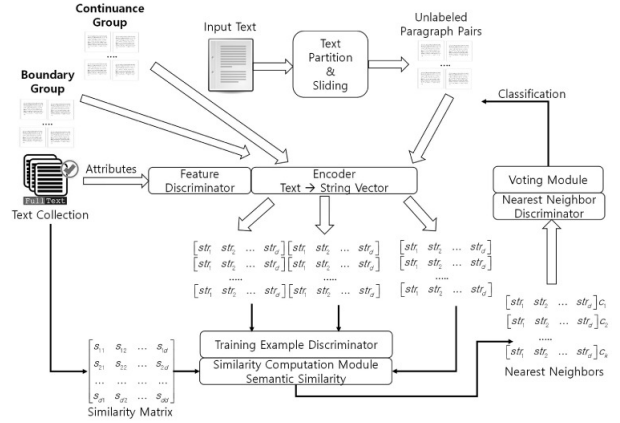


Figure 7. System Architecture

each pair which is classified into boundary in the text, and it is partitioned into subtexts by the boundary.

The execution flow of the text segmentation system is illustrated in Figure 8. Texts with same domain are initially collected, adjacent paragraph pairs are extracted from them, and the paragraph pairs are labeled with boundary or continuance. From an input text, adjacent paragraph pairs are extracted, and are encoded into string vectors, together with the sample paragraph pairs. The adjacent paragraph pairs are classified into boundary or continuance, and there are two groups of paragraph pairs: boundary group and continuance group. The boundary is marked between paragraphs in each pair in the boundary group, and text is partitioned by the marked boundaries into subtexts as the final output of this system.

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The text segmentation is defined as the binary classification where each adjacent paragraph pair is classified into boundary or continuance. The paragraph pairs are encoded into string vectors instead of numerical vectors, and they are classified into directly. The input text is partitioned into subtexts by the boundary which is put between paragraphs which are classified into boundary. The subtexts as the semantic division of the input text are treated as independent texts.

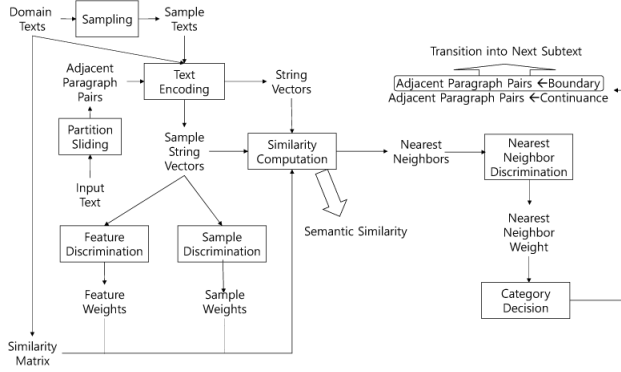


Figure 8. System Flow

D. Connection with Text Classification

This section is concerned with the connection of the text segmentation with the text classification. In the previous section, we mentioned the text segmentation system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section III-B is used for implementing the text classification. This section is intended to describe the connection of the text segmentation system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into string vectors by the process which is described in Section III-A. The similarity computation module is for computing the similarities between string vectors which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

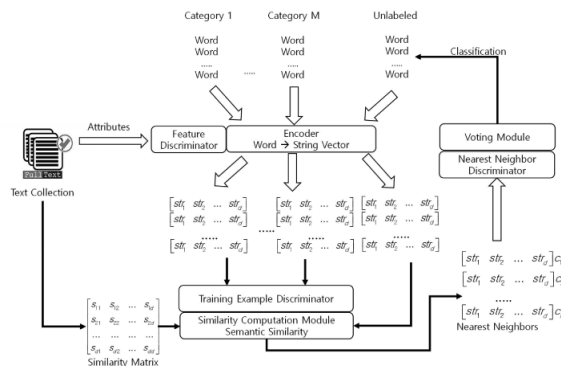


Figure 9. Text Categorization System

The connection of the text segmentation with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the text segmentation system which is described in Section III-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the text segmentation, automatically.

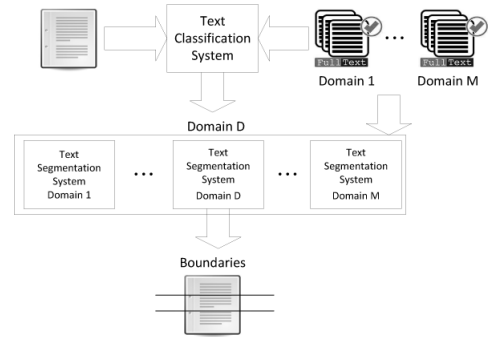


Figure 10. Text Segmentation System connected with Text Categorization

The process of executing the text segmentation, connecting it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its own domain, and sample paragraph pairs each of which is labeled with boundary and continuance are prepared in each domain. The novice text is classified into one among the predefined domains, and the classifier which correspond to the domain is nominated. Adjacent paragraph pairs are generated by sliding the two sized window on the paragraph list, and each paragraph pair is classified into boundary or continuance. Subtexts which are generated by segmenting the text following the boundaries are generated as the final output; the domain which is assigned to the input text is the guide for selecting a classifier.

Let us make some remarks on the connection of the text segmentation with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the text segmentation is to partition the entire text into subtexts based on its content. In the execution flow, the input text is classified into a domain, paragraph pairs are generated by sliding the two sized window, and each paragraph pair is classified into boundary or continuance. We consider connecting the text classification as the main task the text segmentation as the opposite to what is proposed in this research.

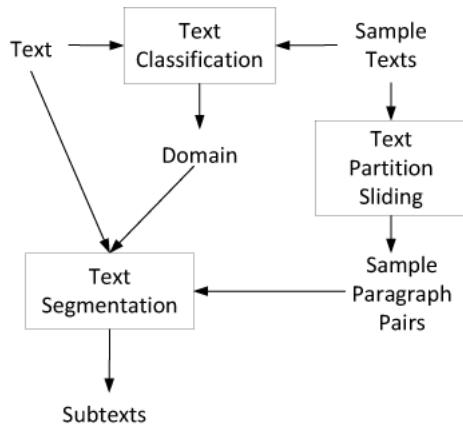


Figure 11. System Flow of Text Segmentation connected with Text Categorization

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We modify the KNN algorithm into the string vector based version. We design the text segmentation system by adopting the proposed KNN algorithm. We connect the system with the text categorization system for automating the domain dependent classification. In the next research, we will implement the text segmentation system as a real system.

Let us mention the remaining tasks for upgrading this research. We define and characterize mathematically more semantic operations on string vectors. Paragraph pairs and texts are encoded into compound representations, each of which consists of more than one structured form. We may modify other machine learning algorithms and deep learning algorithms into string vector-based versions. The text segmentation system which relates to the text classification may be implemented as a real program or a library by adopting the string vector based KNN algorithm.

REFERENCES

- [1] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, International Journal of Information Studies, 2, 2, 2010.
- [2] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, Journal of Network Technology, 1, 1, 2010.
- [3] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, The Proceedings of IEEE/ACS 12th International Conference of Computer Systems and Applications, 2015.
- [4] T. Jo, "String Vector based KNN for Text Categorization", 458-462, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.
- [5] T. Jo, "Long Text Segmentation by String Vector based KNN", 805-809, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.
- [6] T. Jo, "Improving K Nearest Neighbor into String Vector Version for Text Categorization", 1091-1097, ICACT Transaction on Communication Technology, 7, 1, 2018.
- [7] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [8] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, International Journal of Engineering & Technology 7, 1, 2018.
- [9] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.
- [10] T. Jo, "Application of String Vector based K Nearest Neighbor to Content based Segmentation of each News Article", 58-64, The Proceedings of 2nd International Conference on Advanced Engineering and ICT-Convergence, 2019.
- [11] T. Jo, "Application of String Vector based K Nearest Neighbor to Semantic Word Classification", unpublished, 2023.
- [12] T. Jo, "Text Segmentation based on Contents using String Vector based Version of K Nearest Neighbor", in Preparation, 2023.